MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2018/2019

ECP2036 – MICROPROCESSOR SYSTEMS AND INTERFACING

(ME)

25 OCTOBER 2018
9:00 AM – 11:00 AM
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This Question paper consists of 7 pages with 5 questions only.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please write all your answers in the Answer Booklet provided.
4. Opcode map and Special Function Register formats are provided in Appendices.

Question 1

a) What is microcontroller? Name any **FOUR** major internal modules which are available in a microcontroller. [3 marks]

b) Draw the basic connection for a typical 40 pins 8051 microcontroller circuit (with no external memory access). Pin numbers and labels must be clearly indicated in the drawing. [10 marks]

c) What is the effect of system reset on 8051 microcontroller? Explain how the system reset can be performed. [4 marks]

d) Determine the size of address bus and data bus required for the 8051 microcontroller to access an external code memory module with the capacity of 32K*8 bits. [3 marks]

Question 2

a) Analyze the schematic diagram in *Figure Q2* and answer the following questions.

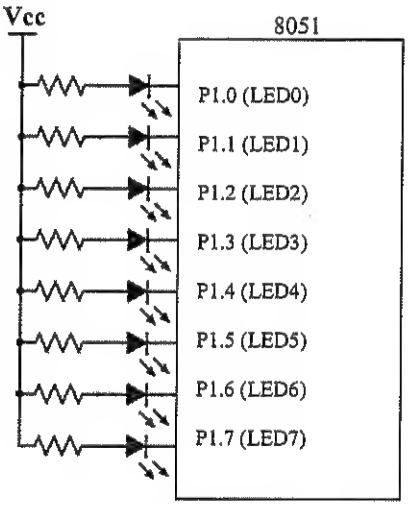

Figure Q2

(i) What is the value should be written to Port1 in order to turn on only LED3 and LED6? [2 marks]

(ii) Write an 8051 assembly program to perform a continuous running LEDs from LED0 to LED7 on Port1 (turn on only one LED at a time for the duration of 1 second). [10 marks]

Continued...

b) Given the following instruction sequence:

Line		Instructions	
01		ORG	0000H
02	MAIN:	MOV	A, #13H
03		ADD	A, #25H
04		DA	A
05		ADD	A, #37H
06		DA	A
07		END	

(i) Convert the instruction sequence into hexadecimal machine language. [4 marks]

(ii) Explain what does the instruction "DA A" as in line 4 and line 6 do? Determine the content of the accumulator after executing line 4 and line 6 respectively. [4 marks]

Question 3

(a) Write an 8051 assembly program to generate 500 Hz square wave from port P1.0 using Timer 0. The generated signal should have a duty cycle of 20%. Find also the adjustments for timer-reload values to obtain high accuracy signal. Assume 12MHz operating frequency. [15 marks]

(b) Determine the names and the values of the special function registers involved for configuring an 8051 serial port to 8-bit UART at 9600 baud rate. Assume 11.0592 MHz operating frequency. [5 marks]

Question 4

(a) List the steps followed by an 8051 microcontroller once it has accepted an interrupt. [6 marks]

(b) Write 8051 instruction(s) to enable the serial interrupt, timer 1 interrupt and external hardware interrupt 0. [4 marks]

(c) P3.3 is connected to an active low push button switch. Every time P3.3 is active low (being pushed), the value in register R1 is incremented by 1. Write an assembly language program using appropriate interrupt to achieve this. [10 marks]

Continued...

Question 5

A numerical keypad with the following arrangement is to be interfaced to an 8051 microcontroller:

'7'	'8'	'9'
'4'	'5'	'6'
'1'	'2'	'3'
'*'	'0'	'#'

(a) Draw a diagram showing the interface using P1 for columns and P2 for rows. [5 marks]

(b) Write a subroutine to continuously scan the keypad and return the position of the row and column of the key pressed into R1 and R2 respectively. [10 marks]

(c) Given the row information is stored in R1, while the column information is stored in R2. Write a subroutine to store the character ('*', '#', '0', '1'...) displayed on the key pressed into the accumulator. [5 marks]

Continued...

Appendix A: Opcode Map

HByte / LByte	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1B, 1C **NOP**	3B, 2C **JBC** bit,rel	3B, 2C **JB** bit,rel	3B, 2C **JNB** bit,rel	2B, 2C **JC** rel	2B, 2C **JNC** rel	2B, 2C **JZ** rel	2B, 2C **JNZ** rel	2B, 2C **SJMP** rel	3B, 2C **MOV** DPTR,#16	2B, 2C **ORL** C,/bit	2B, 2C **ANL** C,/bit	2B, 2C **PUSH** dir	2B, 2C **POP** dir	1B, 2C **MOVX** A,@DPTR	1B, 2C **MOVX** @DPTR,A
1	2B, 2C **AJMP** (P0)	2B, 2C **ACALL** (P0)	2B, 2C **AJMP** (P1)	2B, 2C **ACALL** (P1)	2B, 2C **AJMP** (P2)	2B, 2C **ACALL** (P2)	2B, 2C **AJMP** (P3)	2B, 2C **ACALL** (P3)	2B, 2C **AJMP** (P4)	2B, 2C **ACALL** (P4)	2B, 2C **AJMP** (P5)	2B, 2C **ACALL** (P5)	2B, 2C **AJMP** (P6)	2B, 2C **ACALL** (P6)	2B, 2C **AJMP** (P7)	2B, 2C **ACALL** (P7)
2	3B, 3C **LJMP** addr16	3B, 2C **LCALL** addr16	1B, 2C **RET**	1B, 2C **RETI**	2B, 1C **ORL** dir, A	2B, 1C **ANL** dir, A	2B, 1C **XRL** dir, A	2B, 2C **ORL** C, bit	2B, 2C **ANL** C, bit	2B, 2C **MOV** bit, C	2B, 1C **MOV** C, bit	2B, 1C **CPL** bit	2B, 1C **CLR** bit	2B, 1C **SETB** bit	1B, 2C **MOVX** A, @R0	1B, 2C **MOVX** @R0, A
3	1B, 1C **RR** A	1B, 1C **RRC** A	1B, 1C **RL** A	1B, 1C **RLC** A	3B, 2C **ORL** dir, #data	3B, 2C **ANL** dir, #data	3B, 2C **XRL** dir, #data	1B, 2C **JMP** @A+DPTR	1B, 2C **MOVC** A, @A+PC	1B, 2C **MOVC** A, @A+DPTR	1B, 2C **INC** DPTR	1B, 2C **CPL** C	2B, 1C **CLR** C	2B, 1C **SETB** C	1B, 2C **MOVX** A, @R1	1B, 2C **MOVX** @R1, A
4	1B, 2C **INC** A	1B, 1C **DEC** A	2B, 1C **ADD** A, #data	2B, 1C **ADDC** A, #data	2B, 1C **ORL** A, #data	2B, 1C **ANL** A, #data	2B, 1C **XRL** A, #data	2B, 1C **MOV** A, #data	1B, 4C **DIV** AB	2B, 1C **SUBB** A, #data	1B, 4C **MUL** AB	3B, 2C **CJNE** A, #data, rel	1B, 1C **SWAP** A	1B, 1C **DA** A	1B, 1C **CLR** A	1B, 1C **CPL** A
5	2B, 1C **INC** dir	2B, 1C **DEC** dir	2B, 1C **ADD** A, dir	2B, 1C **ADDC** A, dir	2B, 1C **ORL** A, dir	2B, 1C **ANL** A, dir	2B, 1C **XRL** A, dir	3B, 2C **MOV** dir, #data	3B, 2C **MOV** dir, dir	2B, 1C **SUBB** A, dir	[redacted]	3B, 3C **CJNE** A, dir, rel	2B, 1C **XCH** A, dir	3B, 2C **DJNZ** dir, rel	2B, 1C **MOV** A, dir	2B, 1C **MOV** dir, A
6	1B, 1C **INC** @R0	1B, 1C **DEC** @R0	1B, 1C **ADD** A, @R0	1B, 1C **ADDC** A, @R0	1B, 1C **ORL** A, @R0	1B, 1C **ANL** A, @R0	1B, 1C **XRL** A, @R0	2B, 1C **MOV** @R0, #data	2B, 2C **MOV** dir, @R0	1B, 1C **SUBB** A, @R0	2B, 2C **MOV** @R0, dir	3B, 2C **CJNE** @R0,#data,rel	1B, 1C **XCH** A, @R0	1B, 1C **XCHD** A, @R0	1B, 1C **MOV** A, @R0	1B, 1C **MOV** @R0, A
7	1B, 1C **INC** @R1	1B, 1C **DEC** @R1	1B, 1C **ADD** A, @R1	1B, 1C **ADDC** A, @R1	1B, 1C **ORL** A, @R1	1B, 1C **ANL** A, @R1	1B, 1C **XRL** A, @R1	2B, 1C **MOV** @R1, #data	2B, 2C **MOV** dir, @R1	1B, 1C **SUBB** A, @R1	2B, 2C **MOV** @R1, dir	3B, 2C **CJNE** @R1,#data,rel	1B, 1C **XCH** A, @R1	2B, 2C **XCHD** A, @R1	1B, 1C **MOV** A, @R1	1B, 1C **MOV** @R1, A
8	1B, 1C **INC** R0	1B, 1C **DEC** R0	1B, 1C **ADD** A,R0	1B, 1C **ADDC** A,R0	1B, 1C **ORL** A,R0	1B, 1C **ANL** A,R0	1B, 1C **XRL** A,R0	2B, 1C **MOV** R0, #data	2B, 2C **MOV** dir, R0	1B, 1C **SUBB** A, R0	2B, 2C **MOV** R0, dir	3B, 2C **CJNE** R0,#data,rel	1B, 1C **XCH** A, R0	2B, 2C **DJNZ** R0, rel	1B, 1C **MOV** A, R0	1B, 1C **MOV** R0,A
9	1B, 1C **INC** R1	1B, 1C **DEC** R1	1B, 1C **ADD** A,R1	1B, 1C **ADDC** A,R1	1B, 1C **ORL** A,R1	1B, 1C **ANL** A,R1	1B, 1C **XRL** A,R1	2B, 1C **MOV** R1, #data	2B, 2C **MOV** dir, R1	1B, 1C **SUBB** A, R1	2B, 2C **MOV** R1, dir	3B, 2C **CJNE** R1,#data,rel	1B, 1C **XCH** A, R1	2B, 2C **DJNZ** R1, rel	1B, 1C **MOV** A, R1	1B, 1C **MOV** R1,A
A	1B, 1C **INC** R2	1B, 1C **DEC** R2	1B, 1C **ADD** A,R2	1B, 1C **ADDC** A,R2	1B, 1C **ORL** A,R2	1B, 1C **ANL** A,R2	1B, 1C **XRL** A,R2	2B, 1C **MOV** R2, #data	2B, 2C **MOV** dir, R2	1B, 1C **SUBB** A, R2	2B, 2C **MOV** R2, dir	3B, 2C **CJNE** R2,#data,rel	1B, 1C **XCH** A, R2	2B, 2C **DJNZ** R2, rel	1B, 1C **MOV** A, R2	1B, 1C **MOV** R2,A
B	1B, 1C **INC** R3	1B, 1C **DEC** R3	1B, 1C **ADD** A,R3	1B, 1C **ADDC** A,R3	1B, 1C **ORL** A,R3	1B, 1C **ANL** A,R3	1B, 1C **XRL** A,R3	2B, 1C **MOV** R3, #data	2B, 2C **MOV** dir, R3	1B, 1C **SUBB** A, R3	2B, 2C **MOV** R3, dir	3B, 2C **CJNE** R3,#data,rel	1B, 1C **XCH** A, R3	2B, 2C **DJNZ** R3, rel	1B, 1C **MOV** A, R3	1B, 1C **MOV** R3,A
C	1B, 1C **INC** R4	1B, 1C **DEC** R4	1B, 1C **ADD** A,R4	1B, 1C **ADDC** A,R4	1B, 1C **ORL** A,R4	1B, 1C **ANL** A,R4	1B, 1C **XRL** A,R4	2B, 1C **MOV** R4, #data	2B, 2C **MOV** dir, R4	1B, 1C **SUBB** A, R4	2B, 2C **MOV** R4, dir	3B, 2C **CJNE** R4,#data,rel	1B, 1C **XCH** A, R4	2B, 2C **DJNZ** R4, rel	1B, 1C **MOV** A, R4	1B, 1C **MOV** R4,A
D	1B, 1C **INC** R5	1B, 1C **DEC** R5	1B, 1C **ADD** A,R5	1B, 1C **ADDC** A,R5	1B, 1C **ORL** A,R5	1B, 1C **ANL** A,R5	1B, 1C **XRL** A,R5	2B, 1C **MOV** R5, #data	2B, 2C **MOV** dir, R5	1B, 1C **SUBB** A, R5	2B, 2C **MOV** R5, dir	3B, 2C **CJNE** R5,#data,rel	1B, 1C **XCH** A, R5	2B, 2C **DJNZ** R5, rel	1B, 1C **MOV** A, R5	1B, 1C **MOV** R5,A
E	1B, 1C **INC** R6	1B, 1C **DEC** R6	1B, 1C **ADD** A,R6	1B, 1C **ADDC** A,R6	1B, 1C **ORL** A,R6	1B, 1C **ANL** A,R6	1B, 1C **XRL** A,R6	2B, 1C **MOV** R6, #data	2B, 2C **MOV** dir, R6	1B, 1C **SUBB** A, R6	2B, 2C **MOV** R6, dir	3B, 2C **CJNE** R6,#data,rel	1B, 1C **XCH** A, R6	2B, 2C **DJNZ** R6, rel	1B, 1C **MOV** A, R6	1B, 1C **MOV** R6,A
F	1B, 1C **INC** R7	1B, 1C **DEC** R7	1B, 1C **ADD** A,R7	1B, 1C **ADDC** A,R7	1B, 1C **ORL** A,R7	1B, 1C **ANL** A,R7	1B, 1C **XRL** A,R7	2B, 1C **MOV** R7, #data	2B, 2C **MOV** dir, R7	1B, 1C **SUBB** A, R7	2B, 2C **MOV** R7, dir	3B, 2C **CJNE** R7,#data,rel	1B, 1C **XCH** A, R7	2B, 2C **DJNZ** R7, rel	1B, 1C **MOV** A, R7	1B, 1C **MOV** R7,A

Continued...

Appendix B: Special Function Register Format

TMOD : [Bit 0 (LSB) to Bit 3 is for Timer 0 and Bit 4 to Bit 7 (MSB) is for Timer 1]

GATE	C / /T	M1	MO	GATE	C / /T	MO	MI

GATE: Timer only runs while /INT1 is set.
Cl IT: '1' for event counter, '0' for interval timer
M1, MO: Mode bit select
"00" Mode 0 – 13-bit timer mode
"01" Mode 1 – 16-bit timer mode
"10" Mode 2 – 8-bit auto-reload mode
"11" Mode 3 – Split timer mode

TCON :

TF1	TR1	TFO	TRO	IE1	IT1	IEO	ITO

TCON.7 TF1 Timer 1 overflow flag. Set by hardware on overflow. Clear by hardware when processor vectors to interrupt routine.
TCON.6 TRI Timer 1 run control bit. Set/cleared by software to start/stop timer.
TCON.5 TF0 Timer 0 overflow flag. Set by hardware on overflow. Clear by hardware when processor vectors to interrupt routine.
TCON.4 TRO Timer 0 run control bit. Set/cleared by software to start/stop timer.
TCON.3 IE1 Interrupt 1 Edge flag. Set by hardware when interrupt 1 falling edge is detected. Cleared when interrupt is processed.
TCON.2 IT1 Interrupt 1 Type control bit. Set / cleared by software to specify falling edge / low level triggered external interrupts.
TCON. 1 IEO Interrupt 0 Edge flag. Set by hardware when interrupt 1 falling edge is detected. Cleared when interrupt is processed.
TCON.0 IT0 Interrupt 0 Type control bit. Set / cleared by software to specify falling edge / low level triggered external interrupts.

SCON :

SMO	SM1	SM2	REN	TB8	RB8	TI	RI

SMO	SM1	
0	0	= Shift register mode
0	1	= 8-bit UART mode
1	0	= 9-bit UART mode (Fixed Baud Rate)
1	1	= 9-bit UART mode (Variable Baud Rate)

SM2 = '1' = Enable multiprocessor communication
REN = Receiver Enable
TB8 = Transmit Bit
TI = Transmit Interrupt
RI = Receive Interrupt

Continued...

IE:

EA		ET2	ES	ET1	EX1	ET0	EXO

Bit Position	Symbol	Bit Address	Description
IE.7	EA	AFH	Global enable/disable. EA ='1', each individual source is enable/disable By seetting/clearing its enable bit. EA = 'O', disable all interrupts.
IE.6	-	AEH	Undefined
IE.5	-	ADH	Not implemented in 805 1. ET2 for 8052.
IE.4	ES	ACH	Serial port interrupt enable bit.
IE.3	ET1	ABH	Timer1 interrupt enable bit.
IE.2	EX1	AAH	External interrupt enable bit.
IE. 1	ET0	A9H	Timer0 interrupt enable bit.
IE.0	EXO	A8H	External interrupt enable bit.

IP:

		PT2	PS	PT1	PX1	PTO	PXO

IP.7	-	-	Undefined.
IP.6	-	-	Undefined.
IP.5	-	BDH	Not implemented in 8051. PT2 for 8052.
IP.4	PS	BCH	Serial port interrupt priority bit.
IP.3	PT1	BBH	Timer1 interrupt priority bit.
IP.2	PX1	BAH	External interrupt priority bit.
IP.1	PTO	B9H	Timer-0 interrupt priority bit.
IP.0	PX0	B8H	External interrupt priority bit.

Selected Interrupt Vectors

Interrupt source	Flag	Vector Address
System Reset	RST	0000H
External 0	IEO	0003H
Timer 2 (8052)	TF2 & EXF2	002BH

PSW:

CY	AC	F0	RS1	RSO	OV	-	P

CY : Carry Flag AC : Auxiliary Carry Flag
RS 1, RSO: Register Bank Select
OV : Overflow Flag P : Parity

. End of Paper